Learning of Fuzzy Cognitive Map models without training data

Peer-reviewed author version

NAPOLES RUIZ, Gonzalo; Grau, Isel; CONCEPCION PEREZ, Leonardo; Salgueiro, Yamisleydi & VANHOOF, Koen (2025) Learning of Fuzzy Cognitive Map models without training data. In: Neurocomputing, 623 (Art N° 129409).

DOI: 10.1016/j.neucom.2025.129409
Handle: http://hdl.handle.net/1942/45288

Learning of Fuzzy Cognitive Map models without training data

Gonzalo Nápoles[a,b], Isel Grau[c,d], Leonardo Concepción[b,e], Yamisleydi Salgueiro[f,*], Koen Vanhoof[b]

[a]*Department of Cognitive Science & Artificial Intelligence, Tilburg University, The Netherlands.*
[b]*UHasselt, Business Informatics Research Group, Hasselt, Belgium.*
[c]*Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, The Netherlands.*
[d]*Eindhoven Artificial Intelligence Systems Institute, Eindhoven University of Technology, The Netherlands.*
[e]*Universidad Central de Las Villas, Santa Clara, Cuba.*
[f]*Department of Industrial Engineering, Faculty of Engineering, Universidad de Talca, Campus Curicó, Chile.*

Abstract

This paper introduces a novel zero-data learning algorithm tailored for Fuzzy Cognitive Map (FCM) models utilized in control applications where we must maintain concepts' activation values within predefined intervals. Our approach allows domain experts to specify these intervals and optionally impose weight constraints, ensuring the algorithm produces feasible models. At the core of our approach lies a mathematical formalism that approximates the smallest feasible activation space for each neural concept, which translates into lower and upper bounds for concepts' activation values. Moreover, a parameterized quasi-nonlinear reasoning rule allows controlling whether or not the network converges to a unique fixed point. The learning goal of our algorithm narrows down to computing a weight matrix minimizing the error between the analytical bounds and the target intervals specified by domain experts. To address such a constrained minimization problem, we employ numerical methods operating with approximate gradients, which provide highly accurate solutions with short execution times. The main contribution of our learning algorithm is that it does not require any training data to compute the network structure. Therefore, by accurately approximating the specified activation intervals, our learning algorithm guarantees that the outputs produced by the FCM model will remain within these intervals regardless of the initial conditions used to start the recurrent reasoning process.

Keywords: fuzzy cognitive maps, simulation and control, gradient-based methods, zero-data learning.

1. Introduction

Fuzzy Cognitive Maps (FCMs) [15, 7] are recurrent neural networks where neural concepts represent problem variables while weights characterize the interaction between the concepts. In the traditional FCM formalism, weights encode causal relationships symbolizing the strength and direction in which a concept influences the others. However, modern implementations often replace causal relationships with correlation or association relationships to align the problem domain with the mathematical model and the simulation results.

Unlike other recurrent neural networks, FCMs do not have hidden neurons or layers, providing the model with a degree of transparency [22] while allowing for hybrid reasoning where experts can inject knowledge into the model. When it comes to reasoning, FCM models use a recurrent rule that updates the activation values of neural concepts based on the weight matrix, an activation function, and given initial conditions. Such a recurrent process can be executed synchronously (updating all neurons at once) or asynchronously (updating the neurons in a pre-defined order). A pivotal difference with mainstream recurrent neural networks is that FCMs perform as closed systems, meaning that the neurons' activation values are computed from previously computed values, without receiving fresh data inputs in each iteration. Nonetheless, some interesting extensions reported in [17] and [33] add explicit inputs to the neurons, following the rationale of dynamic systems.

FCM architectures have demonstrated to be a powerful tool for modeling complex systems, where the interactions among concepts reflect the dynamics of the system's behavior. These models allow solving different tasks such as time series forecasting [19, 25, 39, 40], multioutput regression [21, 31], pattern classification [23, 36, 37], scenario analysis [14, 28, 4, 26], and modeling control problems [1, 35]. In particular, when building FCMs for control problems, human expertise and experience are needed. FCM models for control draw on insights from experts who have observed and understood the system's operations and behavior across different conditions.

For example, the authors in [34] presented an FCM architecture devoted to modeling a control problem in a system of tanks and valves. The system contains two tanks with inlet and outlet valves, temperature sensors, and a heating element. The goal of the simulation is to control that the temperature values and the amount of liquid in both tanks range within predetermined values, after activating the heating element. Figure 1 depicts the FCM representing this use case where the relations denote the influence of one concept on another one. For instance *Valve 2* acts as the outlet valve of *Tank 1* and inlet valve of *Tank 2* simultaneously. If the temperature in *Tank 1* increases to the upper

*Corresponding author
Email address: ysalgueiro@utalca.cl (Yamisleydi Salgueiro)

limit by activating the heating element, *Valve 1* opens and fills *Tank 1* with liquid at a low temperature. If, as a consequence, *Tank 1* reaches the upper limit in the amount of liquid, *Valve 2* is opened and liquid is passed from *Tank 1* to *Tank 2*. After several iterations, the system might converge to a stable state where all values of concepts are within the limits and do not change significantly in further iterations.

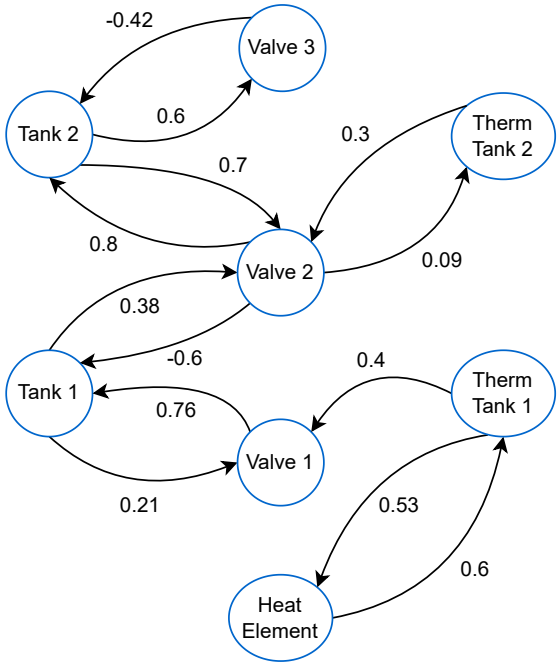


Figure 1: FCM model of the simulation and control use case where the objective is to keep the amount of liquid and the temperature in the tanks within predefined intervals defined by domain experts.

The need to automatically compute the weight matrix of FCM models has motivated the development of learning algorithms. The literature distinguishes between two main approaches: supervised and unsupervised. The first scenario focuses on deriving the weight matrix from a set of input-output data records using an optimization algorithm. Meta-heuristics such as Genetic Algorithms [11, 9, 30], Particle Swarm Optimization [8], and Differential Evolution [10] have been employed in this context. Another optimization strategy relies on regression-like optimizers such as the Moore-Penrose inverse. For example, the FCM architectures in [19, 16] use this strategy to compute the weight matrix that connects the recurrent FCM block with an output layer. The efficiency of this learning approach has motivated other authors to combine it with more elaborated neural architectures involving convolutions or autoencoder layers [12, 41, 6]. Gradient-based optimizers have also proven successful in computing the weight matrix of FCM models [38, 36, 20], as well as other learnable parameters such as the concepts' activation function, especially when domain experts provide the weight matrix [24].

Unsupervised learning algorithms are used for fine-tuning FCM models devoted to simulation and control problems where generalization is not the primary focus. These algorithms typically employ the Hebbian learning principle, which states that if two variables i and j change simultaneously, then the corresponding edge weight should be modified accordingly as $w_{ij} = 1/p \sum_{k=1}^{p} x_i^k x_j^k$, where p is the number of training patterns. Several Hebbian-inspired learning algorithms for FCM models used in control problems can be found in the literature [32, 27, 13], which are still considered the state-of-the-art in this area since error-driven learning algorithm require historical data records. It is worth highlighting that these variants differ from the original Hebbian learning formalism in two ways. On the one hand, they operate on an initial weight matrix, typically provided by domain experts, which may not always be available. On the other hand, they typically rely on a single instance instead of using a training set comprised of several data records, thus leading to FCM models with poor generalization capabilities. Moreover, these learning algorithms often produce weight matrices that lead to unique fixed-point attractors, which produce invariant solutions rather than an acceptable interval of solutions that depend on the initial conditions.

In this paper, we propose a zero-data learning algorithm for FCM models used in control problems where the activation values of decision concepts must be kept within desired intervals. The term *zero-data learning* means that we have no access to a training dataset containing pairs of inputs and outputs (denoting the initial activation values of neural concepts and their expected responses). However, the absence of an explicit training dataset with input-output instances does not imply that the FCM model cannot be trained to perform certain tasks. For example, we can have a comprehensible mathematical description of how the model is expected to behave[1]. In our method, these descriptions are given by domain experts and concern the intervals that contain the activation values of relevant concepts and optionally some weight constraints (e.g., whether the weight between two concepts must be negative, positive, or zero). To compensate for the absence of training data, we introduce a mathematical formalism that computes lower and upper bounds for each concept's activation values at any iteration, using only the information of weights being learned. These bounds are independent of the neurons' initial activation values used to start reasoning, which allows implementing the zero-data learning algorithm. In short, the proposed learning goal consists of computing a weight matrix such that the neurons' analytical bounds are as similar as possible to the intervals specified by the domain experts. Aiming to solve the related constrained optimization task, we resorted to numerical methods that operate with approximate gradients. It is worth stressing that after training is done, the resulting FCM model operates in a crisp fashion where both neurons' activation values and weights are characterized by crisp values rather than intervals, as opposed to what happens with gray FCM extensions.

[1]Despite being conceptually related, zero-data learning should not be confused with zero-shot learning. While the former is a broader concept that implies building a model without any direct data for training, the latter focuses on a model's ability to handle new tasks, categories, or scenarios it has never *explicitly* seen before. However, it is assumed that the model has learned something related that helps it make generalizations.

The rest of the paper is organized as follows. Section 2 introduces the fundamentals concerning fuzzy cognitive mapping together with the relevant mathematical notation. Section 3 outlines the learning problem, emphasizing its formulation as an optimization task. Section 4 discusses the numerical methods selected as optimizers for our proposed approach. Section 5 demonstrates the effectiveness of our proposal through simulations involving randomly generated FCM models and intervals. Finally, Section 6 summarizes the results from the experiments and provides future research directions.

2. Fuzzy Cognitive Maps

Let $\mathbf{C} = \{C_1, \ldots, C_N\}$ a set of N neural concepts in a cognitive network, each denoting a problem variable, and $\mathbf{W}_{N\times N}$ the weight matrix such that $w_{ij} \in [-1, 1]$ characterizes the interaction between the neurons C_i and C_j. For some FCM extensions devoted to multi-output regression, pattern classification, or time series forecasting, such as the Long-term Cognitive Networks [23] or the Long Short-term Cognitive Networks [19], $w_{ij} \in \mathbb{R}$ in order to increase the model's approximation capabilities. In each iteration $t \in \{1, 2, \ldots, T\}$, the network produces an activation vector $\mathbf{A}^{(t)} = (a_1^{(t)}, \ldots, a_i^{(t)}, \ldots, a_N^{(t)})$ where $a_i^{(t)}$ stands for the activation value of the i-th neural concept.

To perform the recurrent reasoning process, an initial activation vector $\mathbf{A}^{(0)} = (a_1^{(0)}, \ldots, a_i^{(0)}, \ldots, a_N^{(0)})$ is provided by the domain expert or elicited from historical data records. Such a vector encodes the initial condition for each neural concept. Eq. (1) shows a quasi-nonlinear reasoning rule [18] used to update the activation vector in each iteration,

$$\mathbf{A}^{(t+1)} = \phi \cdot f\left(\mathbf{A}^{(t)}\mathbf{W}\right) + (1 - \phi) \cdot \mathbf{A}^{(0)} \tag{1}$$

where $\phi \in [0, 1]$ denotes the nonlinearity coefficient and $f(\cdot)$ represents an activation function that prevents neurons' activation values from increasing their absolute magnitude permanently during reasoning. Some popular activation functions reported in the literature are given below.

- *Unipolar activation function*:

$$f_u(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ x, & \text{if } 0 < x < 1 \\ 1, & \text{if } x \geq 1 \end{cases} \tag{2}$$

  This activation function produces values in the $[0, 1]$ interval. It behaves linearly for inputs between 0 and 1, and saturates to 0 for inputs less than or equal to 0, or to 1 for inputs greater than or equal to 1.

- *Bipolar activation function*:

$$f_b(x) = \begin{cases} -1, & \text{if } x \leq -1 \\ x, & \text{if } -1 < x < 1 \\ 1, & \text{if } x \geq 1 \end{cases} \tag{3}$$

  This activation function produces values in the $[-1, 1]$ range such that it behaves linearly for inputs between -1 and 1. It saturates to -1 for inputs less than or equal to -1, or to 1 for inputs greater than or equal to 1.

- *Sigmoid activation function*:

$$f_s(x) = \frac{1}{1 + e^{-\lambda(x-h)}} \tag{4}$$

  When using this function, the neural concepts produce activation values in the $(0, 1)$ range. It behaves as a smooth, sigmoid function, with parameters $\lambda > 0$ and $h \in \mathbb{R}$ controlling the slope and offset of the function.

- *Hyperbolic activation function*:

$$f_h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{5}$$

  This activation function produces values in the $(-1, 1)$ interval. It approximates a linear function for values close to 0, and saturates towards -1 or 1 for large negative or positive inputs, respectively.

In the quasi-nonlinear reasoning rule in Eq. (1), the parameter ϕ allows for controlling the network's convergence to a large extent. More explicitly, if $0 \leq \phi < 1$, it is guaranteed that the FCM model will converge to different fixed points, although cycles could appear if the weight matrix does not fulfill some conditions [18]. If $\phi = 1$, then we will obtain a classic FCM model that might converge to unique fixed points.

The recurrent reasoning rule of an FCM-based model will finish when either (i) the network converges to an equilibrium point, or (ii) a pre-defined maximal number of iterations T is reached. If the network converges, then $(\exists t_\alpha \in \{1, \ldots, (T-1)\} : a_i^{(t+1)} = a_i^{(t)}, \forall i, \forall t \geq t_\alpha)$. In practice, we say that a neural concept has converged if $|a_i^{(t+1)} - a_i^{(t)}| < \xi$ where ξ is typically equal to 1.0e-4. Moreover, these fixed points can be unique (where the neurons produce the same output regardless of the initial conditions) or multiple (where the neurons produce outputs that depend on the initial conditions). Models that fail to converge exhibit limit cycles $(\exists t_\alpha, P \in \{1, \ldots, (T-1)\} : a_i^{(t+P)} = a_i^{(t)}, \forall i, \forall t \geq t_\alpha)$ or chaotic behaviors where neurons' activation values do not follow a consistent pattern.

3. Proposed learning method

In this section, we first introduce relevant mathematical tools supporting our proposal and then we present the learning problem leading to an optimization task.

3.1. Theoretical preliminaries

Before elaborating on the details of weight optimization, we need to estimate the bounds for the activation values of neural concepts in each iteration. Such bounds ensure that these values will always lie within a closed interval, regardless of the initial conditions used to start the reasoning process.

Definition 1. *We say that the closed interval $\mathcal{I}_i^{(t)}$ is the minimal feasible activation space for C_i in the t-th iteration if $\mathcal{I}_i^{(t)}$ is the smallest interval that contains all possible activation values of that neuron for the specified iteration.*

Definition 2. *We say that the closed interval $\mathcal{I}_i$ is the induced activation space for concept C_i if it is the smallest closed interval containing the co-domain of the activation function associated with that neural concept.*

Remark 1. *In Definition 1, considering the initial step, $\mathcal{I}_i^{(0)}$ refers to the induced activation space.*

Definition 3. *We say that C_i is ξ_i-precise in the t-th iteration if* $\sup(\mathcal{I}_i^{(t)}) - \inf(\mathcal{I}_i^{(t)}) \leq \xi_i \leq \sup(\mathcal{I}_i^{(0)}) - \inf(\mathcal{I}_i^{(0)})$.

Remark 2. *We use* $\inf(\mathcal{I}_i^{(t)})$ *and* $\sup(\mathcal{I}_i^{(t)})$ *to represent the bounds (infimum and supremum, respectively) of the closed interval $\mathcal{I}_i^{(t)}$ attached to neuron C_i.*

Definition 4. *We say that C_i is ξ_i-stable if $\exists t_q \in \mathbb{N} : C_i$ is ξ_i-precise in the t-th iteration $\forall t \geq t_q$.*

Remark 3. *Therefore, it holds that $|a_i^{(t)} - a_i^{(t+1)}| \leq \xi_i$ provided that $\xi_i = \sup(\mathcal{I}_i^{(t_q)}) - \inf(\mathcal{I}_i^{(t_q)}), \forall t \geq t_q$.*

The cornerstone of these definitions relies on the minimal feasible activation spaces. While providing an exact computation ensuring these intervals are the smallest ones might be difficult, we can approximate them by only using the weight matrix and the activation function.

The authors in [5] proposed analytical conditions to estimate $\inf(\mathcal{I}_i^{(t+1)})$ and $\sup(\mathcal{I}_i^{(t+1)})$ for FCM models using activation functions that are monotone-increasing and bounded within non-negative intervals. In that regard, they distinguished between two cases: (i) the weight matrix is known but the initial activation values remain unknown, and (ii) only the network topology is known while ignoring the direction and intensity of such relationships and the initial conditions used to start reasoning. In their original work, they excluded some popular functions like the hyperbolic tangent or the rescaled activator. In this section, we extend these formulas to any monotone-increasing and bounded activation function.

3.1.1. Case 1: Known weights, unknown initial conditions

The following equations show how to compute $\inf(\mathcal{I}_i^{(t+1)})$ and $\sup(\mathcal{I}_i^{(t+1)})$ for the i-th neural concept under the assumption that the activation function $f(\cdot)$ is monotone-increasing and bounded within its domain,

$$\inf(\mathcal{I}_i^{(t+1)}) = \phi f(\min(\mathbf{w}_i\mathbf{A}^{(t)})) + (1-\phi)\inf(\mathcal{I}_i^{(0)}) \tag{6}$$

$$\sup(\mathcal{I}_i^{(t+1)}) = \phi f(\max(\mathbf{w}_i\mathbf{A}^{(t)})) + (1-\phi)\sup(\mathcal{I}_i^{(0)}) \tag{7}$$

such that

$$\begin{aligned}\min(\mathbf{w}_i\mathbf{A}^{(t)}) = \frac{1}{2}\sum_{j=1}^{N} w_{ji}\Big(&\sup(\mathcal{I}_j^{(t)})(1-\operatorname{sgn}(w_{ji})) \\ &+ \inf(\mathcal{I}_j^{(t)})(1+\operatorname{sgn}(w_{ji}))\Big)\end{aligned} \tag{8}$$

$$\begin{aligned}\max(\mathbf{w}_i\mathbf{A}^{(t)}) = \frac{1}{2}\sum_{j=1}^{N} w_{ji}\Big(&\sup(\mathcal{I}_j^{(t)})(1+\operatorname{sgn}(w_{ji})) \\ &+ \inf(\mathcal{I}_j^{(t)})(1-\operatorname{sgn}(w_{ji}))\Big)\end{aligned} \tag{9}$$

where $\mathbf{w}_i\mathbf{A}^{(t)}$ denotes the dot product between $\mathbf{A}^{(t)}$ and the i-th column of the weight matrix.

The rationale of these bounds is as follows. When computing $\min(\mathbf{w}_i\mathbf{A}^{(t)})$, if w_{ji} is negative, then the activation value term must be maximal. If w_{ji} is not negative then the activation value term must be minimal. When computing $\max(\mathbf{w}_i\mathbf{A}^{(t)})$, if w_{ji} is negative, then the activation value term must be minimal. If w_{ji} is not negative then the activation value term must be maximal. Moreover, we use the sign function to compute both extreme values within a single equation while avoiding ramifications. The facts that the activation function is monotonically increasing, that ϕ and $1-\phi$ are non-negative numbers and that $\inf(\mathcal{I}_i^{(0)}) \leq a_i^{(0)} \leq \sup(\mathcal{I}_i^{(0)})$ complete the analysis. It should be highlighted that, although we have different conditions for the activation functions, Eq. (8) and Eq. (9) do not change compared to the formalism proposed in [5].

3.1.2. Case 2: Unknown weights, unknown initial conditions

For this case, the estimations for $\inf(\mathcal{I}_i^{(t+1)})$ and $\sup(\mathcal{I}_i^{(t+1)})$ for the i-th neural concept are the same as in Eq. (6) and Eq. (7), respectively. However, the bounds for the dot product between w_i and $A^{(t)}$ are given as shown below:

$$\min(\mathbf{w}_i\mathbf{A}^{(t)}) = -\sum_{j=1}^{N} h_j^{(t)} \tag{10}$$

$$\max(\mathbf{w}_i\mathbf{A}^{(t)}) = \sum_{j=1}^{N} h_j^{(t)}. \tag{11}$$

where $h_j^{(t)} = \max\{|\inf(\mathcal{I}_i^{(t)})|, |\sup(\mathcal{I}_i^{(t)})|\}$.

The rationale for this case is similar to the one in the previous sub-section. To obtain the minimum or the maximum, we always have to pick between $\inf(\mathcal{I}_i^{(t)})$ and $\sup(\mathcal{I}_i^{(t)})$ the one with the highest absolute value. Then, we multiply it by -1 to get the minimum and by 1 to get the maximum.

3.2. Learning problem

Our algorithm is devoted to computing the weight matrix $\mathbf{W}$ ensuring that each concept C_i is ξ-stable as defined in the previous sub-section. More explicitly, the method produces a matrix that guarantees the final activation values of C_i will always lie in the desired interval $[\min_{C_i}, \max_{C_i}]$, regardless of the initial activation values used to start the reasoning process. In this regard, the domain expert is only requested to define desired intervals for each neuronal concept, which can conveniently be arranged in a matrix form as indicated below:

$$\mathbf{Y} = \begin{pmatrix} \min_{C_1} & \max_{C_1} \\ \vdots & \vdots \\ \min_{C_i} & \max_{C_i} \\ \vdots & \vdots \\ \min_{C_N} & \max_{C_N} \end{pmatrix}.$$

Optionally, the expert can specify how the concepts are connected or enforce constraints on the weights to be estimated. This feature is quite relevant since it allows human experts to inject domain knowledge into the model.

Eq. (12) formalizes the constrained minimization problem attached to the proposed learning algorithm,

$$g(\mathbf{W}) = \frac{1}{2}\|\mathbf{Y} - \mathbf{B}\|_2^2 \tag{12}$$

subject to

$$\min_{w_{ij}} \leq w_{ij} \leq \max_{w_{ij}}$$

such that $\mathbf{B}$ contains the infimum and supremum values of each minimum feasible activation space:

$$\mathbf{B} = \begin{pmatrix} \inf(\mathcal{I}_1^{(T)}) & \sup(\mathcal{I}_1^{(T)}) \\ \vdots & \vdots \\ \inf(\mathcal{I}_i^{(T)}) & \sup(\mathcal{I}_i^{(T)}) \\ \vdots & \vdots \\ \inf(\mathcal{I}_N^{(T)}) & \sup(\mathcal{I}_N^{(T)}) \end{pmatrix}.$$

It should be stressed that our proposal does not involve any historical data records since the fitting problem narrows down to computing a weight matrix such that the analytical bounds approximate the desired intervals specified by the domain experts. As a result, our procedure neither requires specifying the initial activation values of neural concepts nor performing the standard reasoning process. Nonetheless, we need to specify in advance which activation function will be used to clip the neurons' activation values and the model's nonlinearity degree, which can be done through the ϕ parameter.

Finally, let us analyze the time complexity of the proposed learning algorithm compared to the traditional approach using K input/output instances. In both cases, we must fine-tune a weight matrix that contains N^2 weights at most. In the traditional approach, every weight is restricted to the $[-1, 1]$ interval, while in our proposal, they can be limited to a narrower interval. The difference between both approaches lies in the evaluation of the error function. In the traditional FCM learning approach, the time complexity of evaluating all training instances is $O(K \cdot N^2 \cdot T)$. In our method, without training data, the time complexity is just $O(N^2 \cdot T)$, which results from the state space estimation procedures introduced earlier in this section. Therefore, the algorithm without training data is less demanding under the assumption that we use the same optimizer to minimize the error function in both cases.

4. Bounded optimizers using approximate gradients

One of the reasons why most learning algorithms for FCM models rarely employ backpropagation through time algorithms is that they often fail due to vanishing and exploding gradient issues. Moreover, gradient-based optimizers require the analytical form of the partial derivatives associated with the loss function, which is often challenging to derive.

However, we can still resort to the two-point finite difference method to approximate the derivative of the loss function at a given point. This numerical method is based on evaluating the function at two points and using the difference in function values to estimate the derivative. Overall, there are different forms of the two-point finite difference method: forward, backward, and central differences. Each form uses a different pair of points to approximate the derivative, as shown below.

- **Forward difference**. It uses the function values at the point of interest w and a point slightly ahead $w+h$, where h is a small step size. The forward difference approximation of the derivative $g'(w)$ is given by:

$$g'(w) \approx \frac{g(w+h) - g(w)}{h} \tag{13}$$

- **Backward difference**. It uses the function values at the point being evaluated w and a point slightly behind $w - h$ to approximate the derivative as follows:

$$g'(w) \approx \frac{g(w) - g(w-h)}{h} \tag{14}$$

- **Central difference**. It uses the function values at points on either side of w, specifically $w+h$ and $w-h$ to approximate the derivative as follows:

$$g'(w) \approx \frac{g(w+h) - g(w-h)}{2h} \tag{15}$$

The error term of the central difference method is $O(h^2)$ compared to $O(h)$ in the other cases, indicating that the error decreases quadratically with h, making it more accurate than the forward and backward differences.

In addition to approximating the derivative of the loss function, we need optimizers supporting bounds. In this paper, we will investigate the learning method's performance using the optimization methods described below.

4.1. The L-BFGS-B method

The limited-memory BFGS with bounds optimizer [3], extensively utilized in machine learning optimization, approximates the inverse Hessian matrix $\mathbf{H}_k$ using a constrained memory allocation. Such an approach renders this optimizer appealing for tackling high-dimensional problems where storing the complete Hessian matrix is not feasible.

The first step consists of computing the gradient $\nabla g(\mathbf{w}_k)$ of the objective function $g(\mathbf{w})$ in each iteration k, such that the search direction $\mathbf{p}_k$ is as follows:

$$\mathbf{p}_k = -\mathbf{H}_k \nabla g(\mathbf{w}_k)$$

where $\mathbf{H}_k$ is the current approximation of the inverse Hessian. A linear search is finds an appropriate step size α_k to update the parameter vector as follows:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{p}_k.$$

Aiming to update the inverse Hessian approximation, the algorithm evaluates the changes relative to both the position and gradient vectors, as depicted below:

$$\mathbf{s}_k = \mathbf{w}_{k+1} - \mathbf{w}_k, \quad \mathbf{y}_k = \nabla g(\mathbf{w}_{k+1}) - \nabla g(\mathbf{w}_k).$$

Using these vectors, $\mathbf{H}_{k+1}$ is updated as follows:

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{s}_k^T \mathbf{y}_k} - \frac{\mathbf{H}_k \mathbf{y}_k \mathbf{y}_k^T \mathbf{H}_k}{\mathbf{y}_k^T \mathbf{H}_k \mathbf{y}_k}.$$

The first term in this equation refers to the Hessian estimated in the previous iteration, while the second term adjusts the approximation using the secant equation, accounting for changes in the parameter vector and gradient. The last term corrects the Hessian approximation to ensure appropriate scaling and normalization, satisfying the curvature condition. By combining these terms the L-BFGS-B algorithm preserves its numerical stability and convergence properties.

4.2. Constrained optimization by linear approximation

The Constrained Optimization BY Linear Approximation (COBYLA) [29] is a numerical optimizer for solving optimization problems that include bound constraints and linear equality or inequality constraints. In a nutshell, this algorithm constructs a sequence of linear approximations to the objective function and constraints. These approximations are optimized within a trust region framework to ensure the constraints are satisfied throughout the optimization process.

Let $g(\mathbf{w})$ be a nonlinear objective function to be minimized subject to some constraints, as depicted below:

$$\begin{aligned} \text{minimize:} \quad & g(\mathbf{w}) \\ \text{subject to:} \quad & h_i(\mathbf{w}) \leq 0, \quad i = 1, 2, \ldots, m \\ & c_j(\mathbf{w}) = 0, \quad j = 1, 2, \ldots, n. \end{aligned}$$

The approach used by COBYLA aims to approximate $g(\mathbf{w})$, $h_i(\mathbf{w})$, and $c_j(\mathbf{w})$ with linear functions around a current point $\mathbf{w}^*$, employing the Taylor series expansion:

$$\begin{aligned} g(\mathbf{w}) &\approx g(\mathbf{w}^*) + \nabla g(\mathbf{w}^*)^T (\mathbf{w} - \mathbf{w}^*) \\ h_i(\mathbf{w}) &\approx h_i(\mathbf{w}^*) + \nabla h_i(\mathbf{w}^*)^T (\mathbf{w} - \mathbf{w}^*) \leq 0 \\ c_j(\mathbf{w}) &\approx c_j(\mathbf{w}^*) + \nabla c_j(\mathbf{w}^*)^T (\mathbf{w} - \mathbf{w}^*) = 0. \end{aligned}$$

After this transformation, we need to minimize the linear approximations of the objective and constraints:

$$\begin{aligned} \text{minimize:} \quad & g(\mathbf{w}^*) + \nabla g(\mathbf{w}^*)^T (\mathbf{w} - \mathbf{w}^*) \\ \text{subject to:} \quad & h_i(\mathbf{w}^*) + \nabla h_i(\mathbf{w}^*)^T (\mathbf{w} - \mathbf{w}^*) \leq 0, \quad i = 1, 2, \ldots, m \\ & c_j(\mathbf{w}^*) + \nabla c_j(\mathbf{w}^*)^T (\mathbf{w} - \mathbf{w}^*) = 0, \quad j = 1, 2, \ldots, n. \end{aligned}$$

At each iteration, COBYLA employs a trust-region strategy to (i) generate a test point, (ii) evaluate the linearized objective and constraint functions at that point, and (iii) adjust the variables to move towards minimizing these linear approximations while enforcing the constraints.

4.3. Sequential quadratic programming

Sequential Quadratic Programming (SLSQP) [2] is a quasi-Newton method devoted to solving nonlinear optimization problems. This iterative numerical optimizer operates under the assumption that the objective function and constraints are twice continuously differentiable, though not necessarily convex. The SLSQP method addresses a sequence of optimization subproblems, each optimizing a quadratic model of the objective function subject to linearized constraints.

SLSQP aims to minimize a quadratic approximation $Q_k(\mathbf{p})$ of the objective function under the constraint $\mathbf{c}(\mathbf{w}_k) + \mathbf{J}_c(\mathbf{w}_k)\mathbf{p} = \mathbf{0}$ such that k denotes the current iteration:

$$\min_{\mathbf{p}} Q_k(\mathbf{p}) \quad \text{subject to} \quad \mathbf{c}(\mathbf{w}_k) + \mathbf{J}_c(\mathbf{w}_k)\mathbf{p} = \mathbf{0}$$

where $\mathbf{p}$ represents the step direction, $\mathbf{w}_k$ is the current iteration point, $\mathbf{c}(\mathbf{w}_k)$ stands for the vector of constraint evaluations at $\mathbf{w}_k$, and $\mathbf{J}_c(\mathbf{w}_k)$ is the Jacobian matrix of constraints evaluated at $\mathbf{w}_k$. In this formalization, the constraint equation $\mathbf{c}(\mathbf{w}_k) + \mathbf{J}_c(\mathbf{w}_k)\mathbf{p} = \mathbf{0}$ ensures that $\mathbf{p}$ moves the current iterate $\mathbf{w}_k$ towards a feasible solution. The quadratic approximation of the objective function is typically expressed as formalized below:

$$Q_k(\mathbf{p}) = g(\mathbf{w}_k) + \nabla g(\mathbf{w}_k)^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \nabla^2 \mathcal{L}(\mathbf{w}_k) \mathbf{p}$$

where $\nabla g(\mathbf{w}_k)$ symbolizes the gradient of the objective function evaluated at the current solution $\mathbf{w}_k$, and $\nabla^2 \mathcal{L}(\mathbf{w}_k)$ stands for the Hessian matrix of the Lagrangian at $\mathbf{w}_k$.

SLSQP combines gradient-based techniques and quadratic programming to search for the optimal solution. This method iterates until it reaches a local minimum where the first-order optimality conditions are sufficiently satisfied. In this regard, SLSQP commonly employs line search methods to ascertain step sizes that ensure a significant reduction in the objective function while adhering to the constraints.

5. Numerical simulations

Before presenting the empirical simulations, we will illustrate the effectiveness of the proposed algorithm for the case study presented in Figure 1. For the sake of verification, we will assume that such a weight matrix is the ideal solution to be found by the optimizer. Since the bounds for this problem are known, the expert would only need to define weight restrictions to be respected during the learning process. In this regard, let $\min_{\mathbf{W}}$ and $\max_{\mathbf{W}}$ be the matrices defining the minimum and maximum values for each weight, which are defined based on the sign of each causal relationship,

$$\min_{\mathbf{W}} = \begin{pmatrix} 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 \\ \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & \epsilon & 0 & 0 \\ 0 & 0 & 0 & 0 & \epsilon & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \epsilon & 0 & 0 & 0 & \epsilon \\ 0 & 0 & 0 & \epsilon & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\max_{\mathbf{W}} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\epsilon & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -\epsilon & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

where $-\epsilon$ and ϵ are small negative and positive values, respectively, used to distinguish non-zero connections from null edges denoting no relationship between concepts.

Table 1 displays the Mean Squared Error (MSE), the Root Mean Squared Error (RMSE), the Mean Absolute Error (MAE), and the Symmetric Mean Absolute Percentage Error (SMAPE) for each gradient-based optimizer. In this experiment, the maximal number of iterations is set to $T = 20$ and the nonlinearity coefficient value is set to $\phi = 0.8$, and the sigmoid activator is used as the activation function. The best-performing algorithm for each metric is highlighted in gray. The reader can notice that both L-BFGS-B and SLSQP provide remarkably accurate approximations for the desired bounds.

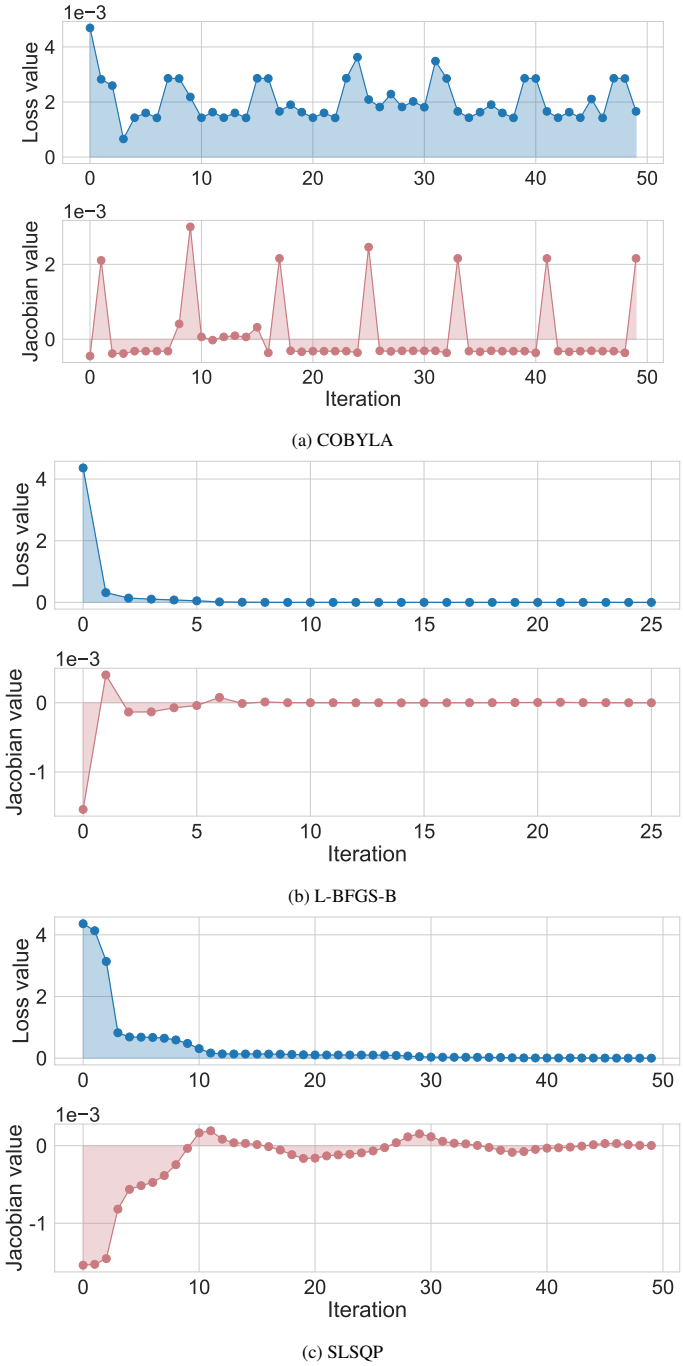
Table 1: Performance metrics reported by each gradient-based optimizer for the control problem depicted in Figure 1.

Algorithm	MSE	RMSE	MAE	SMAPE
COBYLA	6.97e-4	2.64e-2	2.05e-2	3.65e-2
L-BFGS-B	7.78e-10	2.79e-5	1.88e-5	3.55e-5
SLSQP	5.63e-8	2.37e-4	1.14e-4	2.11e-4

Figure 2 depicts the loss function values during the optimization process and the Jacobian values (estimated using Eq. (13)) for the first problem dimension, which corresponds with a null edge. For this example, COBYLA is slightly less effective than L-BFGS-B and SLSQP in handling these constraints. However, it should be stated that null edges should not be optimized in real-world settings since the values of these weights are already known to be zero. However, we decided to include them in this initial experiment to explore the effectiveness of the selected optimizers in extreme situations.

Figure 3 shows the estimated lower and upper bounds for each neuronal concept according to the best-performing algorithm (L-BFGS-B). Moreover, we generated 1000 initial conditions and performed reasoning using the same parametric settings during learning ($T = 20$, $\phi = 0.8$ and $f(x) = \text{sigmoid}$). The activation values produced by the FCM model in each iteration are depicted in gray, providing a visual confirmation of the effectiveness of the theoretical bounds. The reader can notice that the neurons' activation values always lie inside the intervals defined by the lower and upper bounds, regardless of the initial conditions used to start reasoning.

Aiming to conduct more general simulations, we randomly generated 100 FCM models with the number of neurons ranging from 5 to 15 and the edge density from 20% to 80%. Due to the absence of expert knowledge, the constraint for the non-zero edges is given by $-1 \leq w_{ij} \leq 1$ whereas the intervals are obtained with randomly generated initial conditions, thus ensuring that the desired intervals are reachable. It must be stressed that such initial conditions are not used during learning and will not be needed in real-world settings where the experts define the desired intervals. We will retain the parametric settings used in the previous experiment, yet we will explore the algorithm's performance for different activation functions.



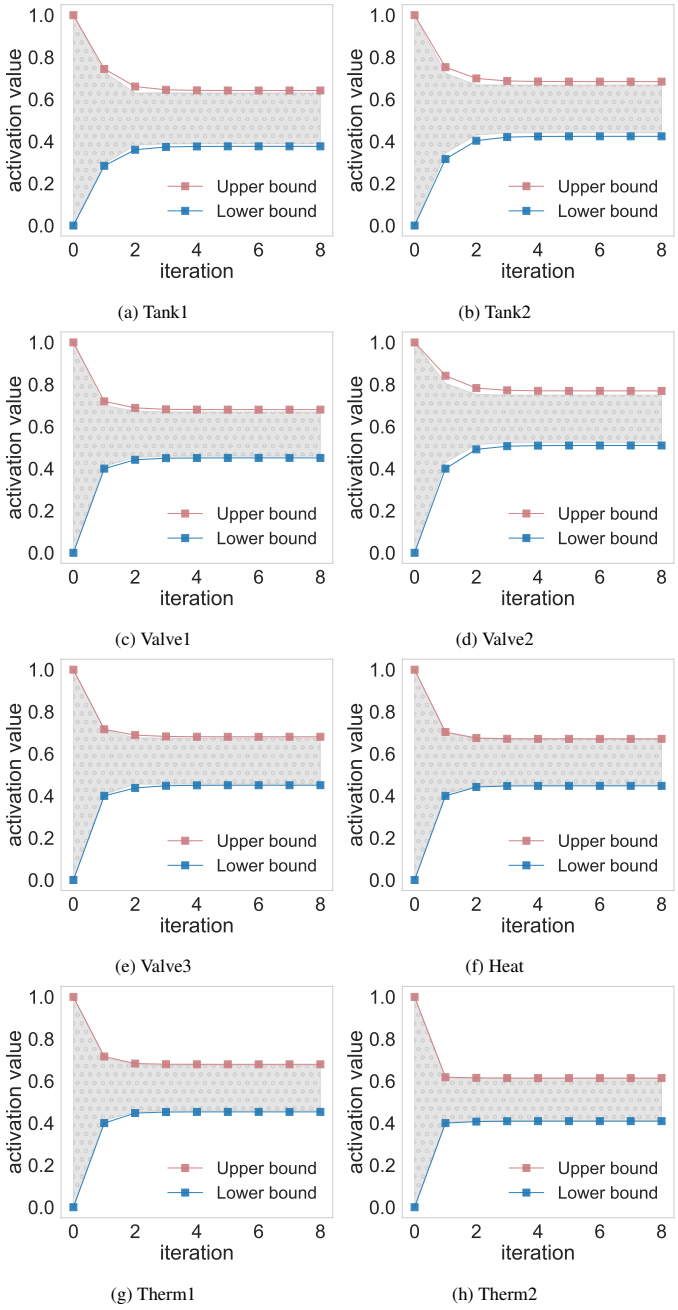
(a) COBYLA

(b) L-BFGS-B

(c) SLSQP

Figure 2: Loss function values and the Jacobian values for a selected dimension for different gradient-based optimizers.

In addition to the bounded optimizers that use approximate gradients, we will experiment with two metaheuristics: Differential Evolution (DE) and Dual Simulated Annealing (DSA).



(a) Tank1 (b) Tank2

(c) Valve1 (d) Valve2

(e) Valve3 (f) Heat

(g) Therm1 (h) Therm2

Figure 3: Upper and lower bounds for each neural concept in the FCM modeling the control problem. The gray area denotes the real activation values in each iteration for 1000 randomly generated initial conditions.

Due to their computational complexity, these optimizers are configured to perform 50 iterations, and neither performs local search operations. The remaining parametric settings of all optimizers employed in our experiments are adopted as specified in their `scipy` implementations.

Tables 2-5 display the performance metrics for the selected optimizers using the saturation unipolar, saturation bipolar, sigmoid, and hyperbolic tangent functions, respectively. Besides reporting the average MSE, RMSE, MAE and SMAPE values, we measure the average training time (expressed in seconds). The best-performing algorithm for each metric is highlighted in gray to facilitate the comparison.

Table 2: Performance metrics for each optimizer (unipolar activator).

Algorithm	MSE	RMSE	MAE	SMAPE	Time
L-BFGS-B	3.9e-5	6.2e-3	2.7e-3	4.0e-3	8.98
COBYLA	1.7e-10	1.3e-5	5.9e-6	9.1e-6	18.86
SLSQP	3.5e-5	5.9e-3	2.6e-3	4.1e-3	7.54
DE	7.5e-6	2.7e-3	1.0e-3	1.6e-3	548.57
DSA	8.3e-4	2.8e-2	1.6e-2	2.3e-2	70.71

Table 3: Performance metrics for each optimizer (bipolar activator).

Algorithm	MSE	RMSE	MAE	SMAPE	Time
L-BFGS-B	7.1e-6	2.6e-3	8.4e-4	1.1e-3	5.8
COBYLA	6.7e-4	2.6e-2	5.8e-3	8.5e-3	15.22
SLSQP	1.1e-3	3.3e-2	1.1e-2	1.3e-2	3.06
DE	1.4e-16	1.2e-8	3.3e-9	4.6e-9	578.69
DSA	9.5e-7	9.7e-4	2.5e-4	3.5e-4	70.94

Table 4: Performance metrics for each optimizer (sigmoid activator).

Algorithm	MSE	RMSE	MAE	SMAPE	Time
L-BFGS-B	7.8e-5	8.8e-3	7.3e-3	2.2e-2	7.56
COBYLA	5.6e-11	7.4e-6	5.2e-6	2.1e-5	58.71
SLSQP	3.3e-3	5.7e-2	4.9e-2	1.2e-1	4.34
DE	1.3e-4	1.1e-2	9.0e-3	2.3e-2	578.76
DSA	1.3e-3	3.6e-2	2.7e-2	7.3e-2	72.63

Table 5: Performance metrics for each optimizer (hyperbolic activator).

Algorithm	MSE	RMSE	MAE	SMAPE	Time
L-BFGS-B	8.9e-5	9.4e-3	7.1e-3	8.4e-3	4.5
COBYLA	1.6e-1	4.1e-1	3.6e-1	4.3e-1	0.97
SLSQP	4.4e-3	6.6e-2	3.9e-2	5.4e-2	2.9
DE	2.1e-5	4.6e-3	3.9e-3	4.8e-3	608.26
DSA	1.1e-4	1.1e-2	8.1e-3	1.1e-2	83.6

When using the unipolar activator, COBYLA emerges as the standout optimizer with low MSE, RMSE, MAE, and SMAPE values, indicating superior precision compared to other algorithms. L-BFGS-B and SLSQP also perform well, reporting competitive results in terms of both accuracy and efficiency. Concerning the metaheuristic methods DE and DSA, while offering reasonable performance in terms of accuracy, they fall behind in terms of runtime efficiency compared to bounded optimizers that approximate gradients.

Concerning the bipolar activation function, the metaheuristic optimizers stand as the clear winner regarding the error metrics. In particular, DE showcases unparalleled precision when approximating the desired intervals through the analytical bounds. However, such efficacy comes at the expense of an unreasonably long execution time, even performing a limited number of iterations. For example, L-BFGS-B produces fairly accurate solutions while being roughly 100 times faster. For this function, SLSQP and SLSQP are less effective.

For the sigmoid activator, COBYLA is the most effective optimizer followed by L-BFGS-B. Although COBYLA produces near-perfect solutions, it is the slowest gradient-based optimizer for this activator. Nonetheless, it continues to be significantly faster than the metaheuristic algorithms, particularly compared to DE. It is noticeable that COBYLA excels when the neurons' activation values lie in the [0, 1] interval.

The results concerning the hyperbolic activation function indicate that DE and L-BFGS-B significantly outperform the other algorithms in terms of MSE, RMSE, MAE, and SMAPE. However, the runtime difference between these two optimizers is overwhelming with the former being about 135 times slower than the latter. In this regard, COBYLA is remarkably fast, but it also accounts for the highest approximation errors. Overall, we recommend the L-BFGS-B method when the neurons' activation values lie in the $[-1, 1]$ interval.

These experiments allow us to draw some partial conclusions. Firstly, theoretical bounds are a powerful tool for approximating the feasible activation intervals of neural concepts without using initial activation values or executing the classic reasoning process. Secondly, both metaheuristics and gradient-based algorithms are able to solve the related optimization problem with high precision, but with varying runtimes. To further explore the trade-off between accuracy and execution time, we will use the following scoring function:

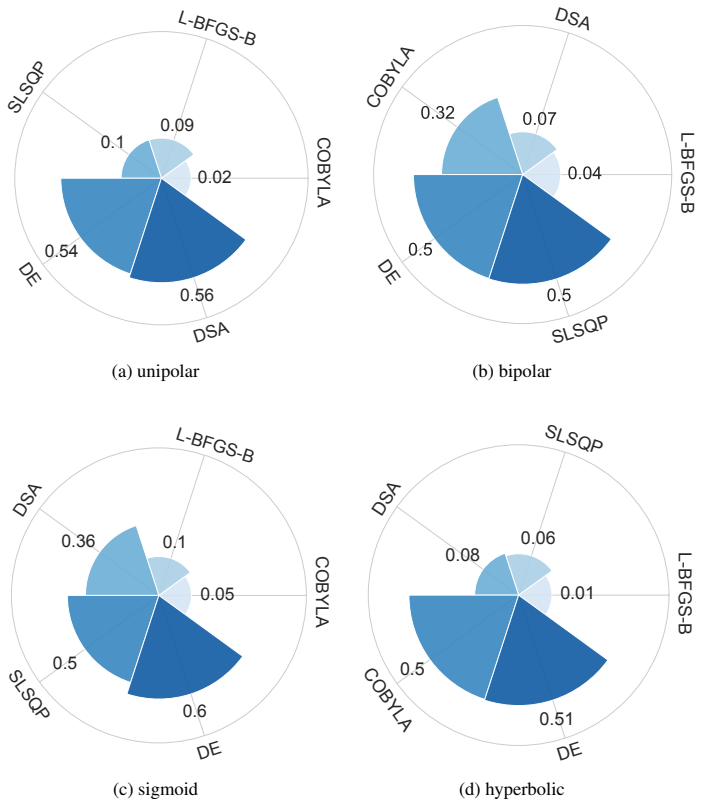
$$\pi_f(o_i) = \frac{1}{2}\left(\frac{e(o_i)}{\max_j\{e(o_j)\}} + \frac{r(o_i)}{\max_j\{r(o_j)\}}\right) \tag{16}$$

such that $e(o_i)$ computes an error metric for the i-th optimizer being evaluated and $r(o_i)$ computes its runtime. Hence, smaller values indicate an effective balance between the accuracy of solutions and the execution time.

Figure 4 ranks the optimizers employed in our study according to the scoring values for each activation function. As a result, the algorithms batter balancing their precision and runtime will be placed at the top of the ranking.

For the unipolar activator, COBYLA is ranked first, followed by L-BFGS-B and SLSQP while DE and DSA are far behind due to their long runtimes. For the bipolar function, L-BFGS-B provides the best compromise between accuracy and execution time, with DE and SLSQP confined to the last positions. Concerning the sigmoid activation function, L-BFGS-B and COBYLA stand as the preferred choices, following the same pattern observed with the unipolar activator, whereas DE and SLSQP once again ranked last. For the hyperbolic activation function, L-BFGS-B and SLSQP reached a fair commitment between efficacy and efficiency, followed by DSA, which ranks third in the performance ranking.

In principle, we could use grid search to select the most suitable optimization method for the problem at hand (as defined by the FCM topology, the constraints, the desired intervals, the activation function, and the maximal number of iterations in the reasoning process). However, based on the simulation results, we recommend using the COBYLA optimizer for unipolar and sigmoid activators and the L-BFGS-B optimizer for the bipolar and hyperbolic tangent functions. Such combinations consistently delivered precise approximations with short runtimes, making it well-suited for control problems requiring a balance between accuracy and computational speed.



(a) unipolar (b) bipolar

(c) sigmoid (d) hyperbolic

Figure 4: Average scores combining the approximation error and runtime for the selected optimizers using different activation functions.

6. Conclusions

In this paper, we presented a zero-data learning algorithm for FCM models used in control problems, specifically designed to ensure that the activation values of selected concepts remain within predefined intervals defined by domain experts. In this regard, we introduced a mathematical formalism that approximates the smallest feasible activation space for any neural concept at any iteration. These bounds depend on the weight matrix but are independent of the initial conditions. By computing these lower and upper bounds for activation values, we can optimize the weight matrix so that the analytical bounds closely match the desired activation intervals. The main contribution of this algorithm to the FCM literature is that it does not require an explicit training dataset composed of input-output instances, as typically happens when training neural architectures using traditional supervised learning approaches.

In our experiments, we used three numerical optimizers supporting constraints and two metaheuristics to solve the related optimization problem. The comparative analysis using synthetically generated datasets and several activation functions revealed that COBYLA and L-BFGS-B are the preferred choices as they reached a fair compromise between their execution time

and the quality of their solutions. The fact that the population-based optimizer reported the longest training time was expected and advocates against their usage in resource-constrained environments. Overall, the simulations confirmed that the proposed algorithm is able to produce high-quality solutions, maintaining output stability regardless of the initial conditions used to perform reasoning. More explicitly, by accurately approximating the specified activation intervals, our learning algorithm guarantees that the outputs produced by the FCM model will always remain within these intervals for any possible initial condition used to perform fresh simulations. This behavior confirms the reliability of the estimated analytical bounds as a replacement for a training dataset containing pairs of inputs and outputs denoting the initial activation values of neural concepts and their expected responses. Nonetheless, it is worth mentioning that the proposed algorithm is devoted to control scenarios where the activation values of selected neural concepts must be bounded. Other applications such as pattern classification, time series forecasting, and multi-output regression would require a labeled training dataset for learning purposes.

Despite the promising results, the proposed algorithm might struggle to provide good approximations if the intervals specified by the expert cannot be produced given the network topology (i.e., without knowledge about the weight matrix or the initial conditions) or if they conflict with each other. While we trust that domain experts can provide reasonable desired bounds, we can assist them in this regard. More explicitly, using Eqs. (6), (7), (10) and (11), we can compute lower and upper bounds for each neural concept without any knowledge about the initial conditions or the weight matrix. In that way, we can ensure that values outside these intervals will never be reachable with any weight matrix $\mathbf{W} \in [-1, 1]^{N \times N}$, which provides valuable knowledge to domain experts. Another research direction consists of exploring the algorithm's performance when historical data is available for only a subset of problem variables. In these cases, we will end up with a hybrid learning algorithm to fine-tune the weight matrix using both the known data measurements for selected concepts and analytical bounds for those for which historical data records do not exist. Finally, we will explore the generation of counterfactual explanations indicating the minimal structural changes to be made to the cognitive network after training leading to a desired change in the analytical bounds of selected concepts.

Acknowledgements

Y. Salgueiro would like to acknowledge the support provided by ANID Fondecyt Regular 1240293 and Basal National Center for Artificial Intelligence CENIA FB210017. L. Concepción was supported by the Special Research Fund (BOF20BL04) from UHasselt, Belgium. Similarly, G. Nápoles would like to acknowledge the support received from the Special Research Fund (BOF24KV18) from UHasselt, Belgium.

References

[1] Ezzeddin Bakhtavar, Mahsa Valipour, Samuel Yousefi, Rehan Sadiq, and Kasun Hewage. Fuzzy cognitive maps in systems risk analysis: a comprehensive review. *Complex & Intelligent Systems*, 7:621–637, 2021.

[2] Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[3] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

[4] Hongyu Chen, Limao Zhang, and Xianguo Wu. Performance risk assessment in public–private partnership projects based on adaptive fuzzy cognitive map. *Applied Soft Computing*, 93:106413, 2020.

[5] Leonardo Concepción, Gonzalo Nápoles, Rafael Falcon, Koen Vanhoof, and Rafael Bello. Unveiling the dynamic behavior of fuzzy cognitive maps. *IEEE Transactions on Fuzzy Systems*, 29(5):1252–1261, 2021.

[6] Guoliang Feng, Wei Lu, and Jianhua Yang. The modeling of time series based on least square fuzzy cognitive map. *Algorithms*, 14(3):69, 2021.

[7] Philippe J Giabbanelli and Gonzalo Nápoles. *Fuzzy Cognitive Maps: Best Practices and Modern Methods*. Springer Nature, 2024.

[8] Petr Hajek and Ondrej Prochazka. Learning interval-valued fuzzy cognitive maps with pso algorithm for abnormal stock return prediction. In Carlos Martín-Vide, Roman Neruda, and Miguel A. Vega-Rodríguez, editors, *Theory and Practice of Natural Computing*, pages 113–125, Cham, 2017. Springer International Publishing.

[9] Petr Hajek and Ondrej Prochazka. Interval-valued fuzzy cognitive maps with genetic learning for predicting corporate financial distress. *Filomat*, 32:1657–1662, 2018.

[10] Shahab Hosseini, Rashed Poormirzaee, Mohsen Hajihassani, and Roohollah Kalatehjari. An ANN-fuzzy cognitive map-based z-number theory to predict flyrock induced by blasting in open-pit mines. *Rock Mechanics and Rock Engineering*, 55:4373–4390, 2022.

[11] William Hoyos, Jose Aguilar, and Mauricio Toro. Prv-fcm: An extension of fuzzy cognitive maps for prescriptive modeling. *Expert Systems with Applications*, 231:120729, 2023.

[12] Georgios D. Karatzinis, Nikolaos A. Apostolikas, Yiannis S. Boutalis, and George A. Papakostas. Fuzzy cognitive networks in diverse applications using hybrid representative structures. *International Journal of Fuzzy Systems*, 25(7):2534–2554, 2023.

[13] Mahsa Khodadadi, Heidarali Shayanfar, Keivan Maghooli, and Amir Hooshang Mazinan. Prediction of stroke probability occurrence based on fuzzy cognitive maps. *Automatika*, 60(4):385–392, 2019.

[14] Kasper Kok. The potential of fuzzy cognitive maps for semi-quantitative scenario development, with an example from brazil. *Global environmental change*, 19(1):122–133, 2009.

[15] Bart Kosko. Fuzzy cognitive maps. *International journal of man-machine studies*, 24(1):65–75, 1986.

[16] Alejandro Morales-Hernández, Gonzalo Nápoles, Agnieszka Jastrzebska, Yamisleydi Salgueiro, and Koen Vanhoof. Online learning of windmill time series using long short-term cognitive networks. *Expert Systems with Applications*, 205:117721, 2022.

[17] Vassiliki Mpelogianni and Peter P Groumpos. Re-approaching fuzzy cognitive maps to increase the knowledge of a system. *Ai & Society*, 33:175–188, 2018.

[18] Gonzalo Nápoles, Isel Grau, Leonardo Concepción, Lisa Koutsoviti Koumeri, and João Paulo Papa. Modeling implicit bias with fuzzy cognitive maps. *Neurocomputing*, 481:33–45, 2022.

[19] Gonzalo Nápoles, Isel Grau, Agnieszka Jastrzebska, and Yamisleydi Salgueiro. Long short-term cognitive networks. *Neural Computing and Applications*, 34(19):16959–16971, 2022.

[20] Gonzalo Nápoles, Agnieszka Jastrzebska, Isel Grau, and Yamisleydi Salgueiro. Backpropagation through time learning for recurrence-aware long-term cognitive networks. *Knowledge-Based Systems*, 295:111825, 2024.

[21] Gonzalo Nápoles, Agnieszka Jastrzebska, Carlos Mosquera, Koen Vanhoof, and Władysław Homenda. Deterministic learning of hybrid fuzzy cognitive maps and network reduction approaches. *Neural Networks*, 124:258–268, 2020.

[22] Gonzalo Nápoles, Nevena Ranković, and Yamisleydi Salgueiro. On the interpretability of fuzzy cognitive maps. *Knowledge-Based Systems*, 281:111078, 2023.

[23] Gonzalo Nápoles, Yamisleydi Salgueiro, Isel Grau, and Maikel Leon Espinosa. Recurrence-aware long-term cognitive network for explainable pattern classification. *IEEE Transactions on Cybernetics*, 53(10):6083–6094, 2023.

[24] Gonzalo Nápoles, Frank Vanhoenshoven, Rafael Falcon, and Koen Vanhoof. Nonsynaptic error backpropagation in long-term cognitive networks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(3):865–875, 2020.

[25] Omid Orang, Petrônio Cândido de Lima e Silva, and Frederico Gadelha Guimarães. Time series forecasting using fuzzy cognitive maps: a survey. *Artificial Intelligence Review*, 56:7733–7794, 2023.

[26] Bernardo MR Paiva, Fernando AF Ferreira, Elias G Carayannis, Constantin Zopounidis, João JM Ferreira, Leandro F Pereira, and Paulo JVL Dias. Strategizing sustainability in the banking industry using fuzzy cognitive maps and system dynamics. *International Journal of Sustainable Development & World Ecology*, 28(2):93–108, 2021.

[27] G. A. Papakostas, D. E. Koulouriotis, A. S. Polydoros, and V. D. Tourassis. Towards hebbian learning of fuzzy cognitive maps in pattern classification problems. *Expert Systems with Applications*, 39:10620–10629, 2012.

[28] Inês PC Pereira, Fernando AF Ferreira, Leandro F Pereira, Kannan Govindan, Ieva Meidutė-Kavaliauskienė, and Ricardo JC Correia. A fuzzy cognitive mapping-system dynamics approach to energy-change impacts on the sustainability of small and medium-sized enterprises. *Journal of Cleaner Production*, 256:120154, 2020.

[29] M. J. D. Powell. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pages 51–67. Springer Netherlands, 1994.

[30] Alexander Rotshtein, Brian A Polin, Denys I Katielnikov, and Neskorodieva Tetiana. Modeling of russian–ukrainian war based on fuzzy cognitive map with genetic tuning. *The Journal of Defense Modeling and Simulation*, 0(0):15485129231184900, 0.

[31] Alexander Rotshtein, Arthur Yosef, Tatiana Neskorodeva, and Denys Katielnikov. Fuzzy cognitive map and mean square method in empirical modeling: Application in economics. *Expert Systems with Applications*, 247:123176, 2024.

[32] Jose L. Salmeron and Pedro R. Palos-Sanchez. Uncertainty propagation in fuzzy grey cognitive maps with hebbian-like learning algorithms. *IEEE Transactions on Cybernetics*, 49(1):211–220, 2019.

[33] Richar Sosa, Alejandro Alfonso, Gonzalo Nápoles, Rafael Bello, Koen Vanhoof, and Ann Nowé. Synaptic learning of long-term cognitive networks with inputs. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.

[34] Chrysostomos D Stylios and Peter P Groumpos. Fuzzy cognitive maps: a model for intelligent supervisory control systems. *Computers in Industry*, 39(3):229–238, 1999.

[35] Chrysostomos D Stylios and Petros P Groumpos. Modeling complex systems using fuzzy cognitive maps. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 34(1):155–162, 2004.

[36] Piotr Szwed. Classification and feature transformation with fuzzy cognitive maps. *Applied Soft Computing*, 105:107271, 2021.

[37] Marios Tyrovolas, X. San Liang, and Chrysostomos Stylios. Information flow-based fuzzy cognitive maps with enhanced interpretability. *Granular Computing*, 8:2021–2038, 2023.

[38] Jingyuan Wang, Zhen Peng, Xiaoda Wang, Chao Li, and Junjie Wu. Deep fuzzy cognitive maps for interpretable multivariate time series prediction. *IEEE Transactions on Fuzzy Systems*, 29(9):2647–2660, 2021.

[39] Yihan Wang, Fusheng Yu, Wladyslaw Homenda, Witold Pedrycz, Yuqing Tang, Agnieszka Jastrzebska, and Fang Li. The trend-fuzzy-granulation-based adaptive fuzzy cognitive map for long-term time series forecasting. *IEEE Transactions on Fuzzy Systems*, 30(12):5166–5180, 2022.

[40] Kai Wu, Jing Liu, Penghui Liu, and Shanchao Yang. Time series prediction using sparse autoencoder and high-order fuzzy cognitive maps. *IEEE Transactions on Fuzzy Systems*, 28(12):3110–3121, 2020.

[41] Shanchao Yang and Jing Liu. Time-series forecasting based on high-order fuzzy cognitive maps and wavelet transform. *IEEE Transactions on Fuzzy Systems*, 26(6):3391–3402, 2018.